Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 28, 2015

D. Zhang
Huawei

October 25, 2014

Certificate Transparency for Domain Name System Security Extensions
draft-zhang-trans-ct-dnssec-00

Abstract

In draft-ietf-trans-rfc6962-bis, a solution is proposed for publicly logging the existence of Transport Layer Security (TLS) certificates using Merkle Hash Trees. This document tries to use this idea in DNSSEC and publicly logging the DS RRs in order to notice the issuance of suspect key signing keys.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 28, 2015.

Table of Contents

1. Introduction

[I-D.ietf-trans-rfc6962-bis] specifies a Certificate Transparency (CT) mechanism to disclosing TLS certificates into public logs so as to benefit the public to monitor the operations in issuing certificates to improper subscribers. The logs do not prevent mis-issuing behavior directly, but the provided public audibility can increase the possibility in detecting the improper behaviors of issuers. The logs are constructed with Merkle Hash Trees to ensure the append-only property, and thus enable anyone to verify the correctness of each log. Note that CT is a common mechanism although

[I-D.ietf-trans-rfc6962-bis] only specify how to use it to publish TLS server certificates issued by public certificate authorities (CAs).

This document discusses the application of CT in addressing the improper issuance issues in DNSSEC. DNSSEC establishes chains of public keys for clients to assess the validity of DNS resource records. In order to prove the validity of keys used for signing DNS data, DNSSEC uses DNS public key (DNSKEY) RRsets and Delegation Signer (DS) RRsets to form authentication chains for the signed data, with each link in the chains vouching for the next by signing the next. If an authentication chain can be eventually connected to the a trusted DNS key or DS RR, the client can then ensure the key for signing the data is legitimate. Unlike PKIX, SDNSEC inherently has strong naming constraints. The owner of a zone can only be allowed to sign the RRs in his zone. Any attempt in signing the RRs in other zones will be easily detected by clients. However, the owner of a zone is dependent on its parent delegation via the DS record to vouch for its DNSKEY. The zone itself is responsible for publishing DS records for the child zones that dependant on it. Misbehavior or compromise of the parent zone directly affects the core DNS security of the child zone. A detailed example is provided in Section 3.

In order to benefit the detection of improper issuance/delegation of DNSSEC keys, this document describes an extension to CT to support logging DSs . The CT logs are publicly auditable, making it possible for anyone to verify the correctness of the log entries and monitor the new DS RR's appended to the log. The logs do not prevent the parent from issuing DS records that the child disagrees with, but they ensure that interested parties can detect such operations. For instance, For example, a zone owner that has been compromised or compelled by a third party can hijack a child zone to return different DNS data that is indistinguishable from DNSSEC validated data from the child zone by using its own DNSKEY to sign DNS data on behalf of the child zone. It could deliver this modified DNS data to only selected regions or individuals, making this attack very difficult to detect by the legitimate child zone.

In DNSSEC, it is assumed that the keys used for signing RRs or other keys will be properly maintained. This work follows this assumption and the compromise of key signing keys are out of scope of this work. This work assumes the existence of inside attacker. That is, a legal owner of a zone may try to attack or circumvent other zones. However, because the naming constraint feature of DNSSEC, a zone owner in principle can only use its keys to perform attacks on its child zones.

This work reuses most of the messages and data structures specified in [I-D.ietf-trans-rfc6962-bis] and makes necessary extensions for supporting DS RRs. Only the extensions to [I-D.ietf-trans-rfc6962-bis] are presented in this document.

2. Cryptographic Components of Certificate Transparency

The introduce of cryptographic components of CT is in Section 2 of [I-D.ietf-trans-rfc6962-bis]. When applying CT for NDSSEC, a log is a single, ever-growing, append-only Merkle Tree of DS RRs.

3. Motivation Scenario

Assume a zone (foo.bar.example) and its parent zone (bar.example) are owned by different organizations. Follows are the steps of an example attack that the owner of the parent zone could perform on the child zone.

1. Set up a fake foo.bar.example DNS server

2. It generates a new key signing key X1 and zone signing key X2. It uses the KSK to sign the ZSK. It uses the ZSK to sign its resource records.

3. It generates a DS record for the KSK record it generated in step 2.

4. The owner of bar.example sign the DS RR with its zone signing key and publishes it

5. Change the IP address of the DNS server of foo.bar.example in the associated RRs to the IP address of the fake DNS server

The owner of foo.bar.example may try to periodically access the DNS server of bar.example and monitor the RRs on it . However, there is still a time window between two assessments which can be taken advantage of by the owner of bar.example to perform a hijacking attack and remove the bogus RRs before the owner of foo.bar.example detects the attack.

In some cases, the parent can even achieve its objectives without publishing the DS RR, which makes the attacks more difficult to detect.

If the owner of bar.example is forced to publish his operations on the public CT logs, any improper behaviors will be detected eventually. Through checking the log, it is easy detect the improper issuance of RRs of his parent zone.

4. Log Format and Operation

As illustrated in Section 3, a zone owner may need to publish multiple RRs in order to hijack the queries to its child zone and re-direct them to another illegal DNS server. However, it is not necessary to publish all those associated RRs to the log. In fact, by publishing the DS RR which is critical in constructing the authentication chain across two zones will be sufficient for helping the public to detect the improper issuance behavior. In this solution, when a zone owner generates a DS RR and delegates a new public key to a child zone, it MUST publish the DS RR at least one CT log in order to allow the public to monitor its behavior. Identical to what is specified in [I-D.ietf-trans-rfc6962-bis], each CT log needs to return a SCT to the zone owner immediately. The SCT will be encapsulated in a SCT RR and published within a DS RR.

The SCT is the log's promise to incorporate the RR in the Merkle Tree within a fixed amount of time known as the Maximum Merge Delay (MMD). If the log has previously seen the certificate, it MAY return the same SCT as it returned before. DNS servers MUST provide an SCT within a SCT RR. DNSSEC clients will not honor a DS RR that does not have a valid SCT. Therefore it is expected that a zone owner will usually deliver the DS RRs for audit purposes.

4.1. Log Entries

A zone owner can submit a DS RR to any preferred logs before publishing the RR. In order to enable attribution of each logged RR to its issuer, the log SHALL publish a list of acceptable public keys (or hashes of public keys) of root zone or islands of security. Each submitted DS RR MUST be accompanied by all additional RRs (DNSKEY RRs, DS RRs, and RRSIG RRs) which construct an authentication chain to an accepted root public key.

Logs MUST verify that the authentication chain and make sure it leads back to a trusted public key, using the chain of intermediate DNSKEY RRs and DS RRs provided by the submitter. Logs MUST refuse to publish a DS RR without a valid chain to a trusted key. If a DS RR is accepted and an SCT issued, the accepting log MUST store the entire chain used for verification, including the DS RR itself and including the trusted key used to verify the chain, and MUST present this chain for auditing upon request.

To comply with the certificate entries specified in [I-D.ietf-trans-rfc6962-bis],Each DS RR entry in a log MUST include the following components:

```
    enum { x509_entry(0), precert_entry(1), DSRR_entry(TBD1),(65535) }
LogEntryType;

    struct {
        LogEntryType entry_type;
        select (entry_type) {
            case x509_entry: X509ChainEntry;
            case precert_entry: PrecertChainEntry;
            case DSRR_entry:DSRR_Chain_Entry
        } entry;
    } LogEntry;

    opaque DNSSECRR<1..2^24-1>;

    struct {
       DNSSECRR DSRR;
       DNSSECRR DNSSEC_key_chain<0..2^24-1>
    } DSRR_Chain_Entry;
```

"entry_type" is the type of this entry. the type value of a DSRR LogEntry is TBD.

"DSRR" is the DS RR submitted for auditing.

"DNSSEC_key_chain" is a chain of additional DNSSEC RRs required to verify the DS RR.A typical authentication chain is as follow: Trusted DNSSKEY ->[DS->(DNSKEY)*->DNSKEY]*-> Submitted DS RR, where "*" denotes zero or more sub-chains. (DNSKEY)* indicates that DNSSEC permits additional layers of DNSKEY RRs including the keys for signing other keys within a zone. Each DNSKEY/DS RR in the chain is authenticated by a RRSIG RR. In practice, a RRSIG RR is normally used to sign a DS/DNSKEY RRset. Therefore, not only the DS/DNSKEY RR on the authentication chain but also other records in the RRset SHOULD be provided to the log the verification purpose. Otherwise, the log may have to consult DNS again in order to verify the authentication chains. Logs SHOULD limit the length of chain they will accept.

4.2. Structure of the Signed Certificate Timestamp

This work reuses the structure of Signed Certificate Timestamp specified in Section 3.3 of [I-D.ietf-trans-rfc6962-bis] but make necessary extensions.

```
    enum { certificate_timestamp(0), tree_hash(1),DSRR_timestamp(TBD2), (255) }
      SignatureType;

    enum { v1(0), (255) }
      Version;

      struct {
          opaque key_id[32];
      } LogID;

     struct {
        opaque issuer_key_hash[32];
        C14N_DSRR dsrr;
      } DSRR;

      opaque CtExtensions<0..2^16-1>;
```

"key_id" and "issuer_key_hash" are defined in Section 3.3 of [I-D.ietf-trans-rfc6962-bis].

dsrr is the submitted DS RR in a canonical form. The canconicalization of a DS RR is described in Section 6.2 of [RFC4304].

```
       struct {
           Version sct_version;
           LogID id;
           uint64 timestamp;
           CtExtensions extensions;
           digitally-signed struct {
               Version sct_version;
               SignatureType signature_type = DSRR_timestamp;
               uint64 timestamp;
               LogEntryType entry_type;
               select(entry_type) {
                   case x509_entry: ASN.1Cert;
                   case precert_entry: PreCert;
                   case BIN_entry: BinaryDigest;
                   case BINDI_entry: BinaryDigest
               } signed_entry;
              CtExtensions extensions;
           };
       } SignedCertificateTimestamp;
```

The encoding of the digitally-signed element is defined in [RFC5246].

"sct_version", "timestamp", "entry_type and extensions" are are identical to what is defined in Section 3.3 of [I-D.ietf-trans-rfc6962-bis]..

"signed_entry" is the is DSRR (in the case of a DSRR_entry), as described above.

"extensions" are future extensions to this protocol version (v1). Currently, no extensions are specified.

4.3. Merkle Tree

This specification extends teh structure of the Merkle Tree input in Section 3.5 of [I-D.ietf-trans-rfc6962-bis] and enable it to encapsulate DS RR:

```
enum { v1(0), v2(1), (255) }
  LeafVersion;

struct {
    uint64 timestamp;
    LogEntryType entry_type;
    select(entry_type) {
        case x509_entry: ASN.1Cert;
        case precert_entry: PreCert;
        case DSRR_entry: DSRR;
    } signed_entry;
    CtExtensions extensions;
} TimestampedEntry;

struct {
    LeafVersion version;
    TimestampedEntry timestamped_entry;
} MerkleTreeLeaf;
```

The fields in the input are introduced in Section 3.5 of [I-D.ietf-trans-rfc6962-bis].

Open question[dacheng]: We should include the RRs constucting the authenticaiton chain in the input, right?

5. Including the Signed Certificate Timestamp into DNS Security Extensions

In section 3.5 of [I-D.ietf-trans-rfc6962-bis]

5.1. SCT RR

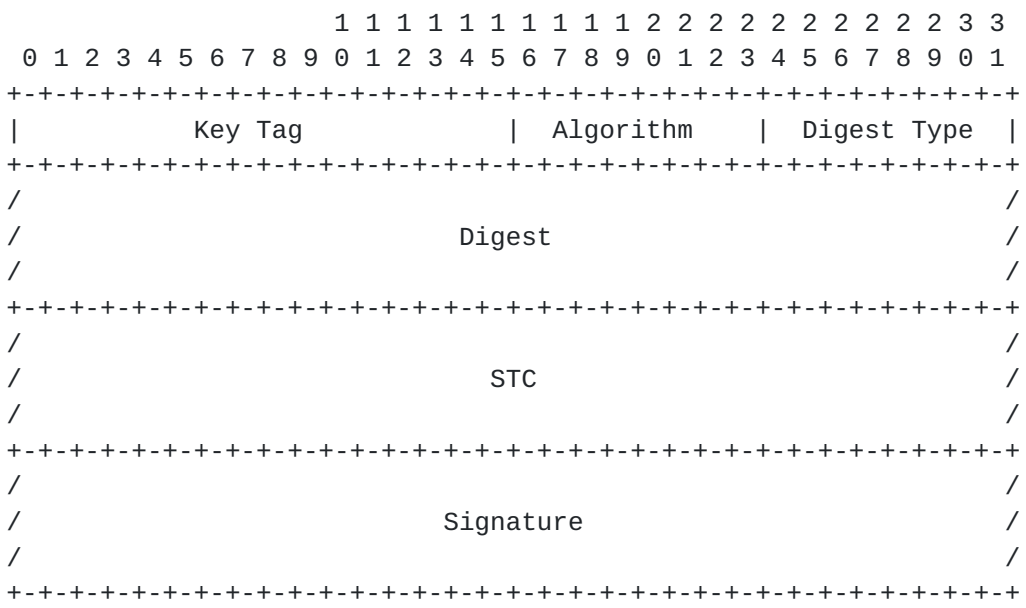
The SCT associated with a DS RR is stored within a STC RR. A DNS server MAY provide multiple SCT RRs for one DS RR.

The type number for the SCT record is TBD3.

The SCT resource record is class independent.

The life period of SCT RR should not be set in a way that the RR will not be expired before the associated DS RR.

The RDATA portion of an SCT RR is as shown below.



5.1.1. The Key Tag Field

The Key Tag field lists the key tag of the DNSKEY RR referred to by the SCT record, in network byte order. Appendix B of [RFC4034] describes how to compute a Key Tag.

5.1.2. The Algorithm Field

The Algorithm field lists the algorithm number of the DNSKEY RR referred to by the SCT record. Appendix A.1 of [RFC4034] lists the algorithm number types.

5.1.3. The Digest Type Field

The Digest Type field identifies the algorithm used to construct the digest used to identify the DS RR that the SCT RR refers to. Appendix A.2 of [RFC4034] lists the possible digest algorithm types.

5.1.4. The Digest Field

The method of calculating digest is identical to what is specified in Section 5.1.4 of [RFC2065].[RFC4034]

5.1.5. The SCT Field

This field contains the SCT got from the log, encoded in BASE64.

5.1.6. The Signature Field

This field contains the SCT signature associated with the SCT. The Signature field is represented as a Base64 encoding of the signature.

5.2. Operations

After introducing the SCT RR, the verification procedures of DNS data specified in DNSSEC[RFC4305] do not change a lot. However, the correctness of CTS needs to be assessed during checking the validity of a DS RR.

A DS RR needs to be associated with a CTS RR which contains a valid CTS and signed with a proper public key. Otherwise, the DS RR will not be used to construct the authentication chain. The signatures of DS RR and its CTS RR should be stored in different RRSIG RR respectively. In addition, a DNS server will sends CTS RRs and the associated RRSIG RRs to a resolver only when it indicates the support of CT in the request.

6. Log Client Messages

In Section 4 of [I-D.ietf-trans-rfc6962-bis], a set of messages is defined for clients to query and verfiy the correctness of the log entries they are interested in. In this memo, two new messages are defined for CT to support DNSSEC.

6.1. Add DNSSEC RR Chain to Log

POST https://<log server>/ct/v1/add-RR-chain

Inputs:

chain: An array of base64-encoded DNS RR. The first element is the submited DS RR; the second chains to the first and so on to the last, which is a trurst DNSKey RR.

Outputs:

sct_version: The version of the SignedCertificateTimestamp structure, in decimal. A compliant v1 implementation MUST NOT expect this to be 0 (i.e., v1).

id: The log ID, base64 encoded.

timestamp: The SCT timestamp, in decimal.

extensions: An opaque type for future expansion. It is likely that not all participants will need to understand data in this field. Logs should set this to the empty string. Clients should decode the base64-encoded data and include it in the SCT.

signature: The SCT signature, base64 encoded.

6.2. Retrieve Accepted Root DNSKEY RRs

GET https://<log server>/ct/v1/get-root-RRs

No inputs.

Outputs:

RRs: An array of base64-encoded DNSKEY RRs that are acceptable to the log.

7. IANA Considerations

8. Security Considerations

8.1. Logging other types of RRs

This solution only tries to describes a solution to disclose keys for DNSSEC in logs for the public to audit. However, it may be valuable to also log the RRs specified in [RFC1035]. For instance, assume there is an attacker which has compromised the zone authentication key and is able to perform the MITM attack between a resolver and the

DNS server of the zone. It is possible for an attacker to transfer a forged RR which is signed with the compromised key. The current solution cannot benefit the detection of this attack in this scenario. However, if the RR is also required to be uploaded to public logs, the condition is changed. If the attacker does not publish the RR to a log, it cannot get the SCT. When the attacker tries to publish the RR to the log, the owner of the zone may detect the problem even if the attacker can provide keys to convince the log to accept the RR.

8.2. Scalability Concerns

The log MAY limit accepting entries where the TTL is too short or the RRSIG times are too far in the future or the past, to avoid spamming the log. It should probably also put a maximum on the number of child zones to avoid getting spammed.

9. Acknowledgements

10. Normative References

[I-D.ietf-trans-rfc6962-bis]
Laurie, B., Langley, A., Kasper, E., and R. Stradling, "Certificate Transparency", draft-ietf-trans-rfc6962-bis-04 (work in progress), July 2014.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2065] Eastlake, D. and C. Kaufman, "Domain Name System Security Extensions", RFC 2065, January 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4304] Kent, S., "Extended Sequence Number (ESN) Addendum to IPsec Domain of Interpretation (DOI) for Internet Security Association and Key Management Protocol (ISAKMP)", RFC 4304, December 2005.

[RFC4305] Eastlake, D., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4305, December 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Author's Address

Dacheng Zhang
Huawei

Email: zhangdacheng@huawei.com